Protecting Train Balise Telegram Data Integrity*

Huaqun Guo, Senior Member, IEEE, Jonathan Zhi Wei Sim, Bharadwaj Veeravalli, Jiqiang Lu

Abstract—ETCS (European Train Control System) is an automatic control system that controls the speed limits of a train. Correct operation of this system is crucial as malfunctions in the system can cause serious accidents such as train collisions. It is essential that train-ground communication is reliable as this ensures the smooth operation of trains and accurate train parking on stations. The ground-based balise is a passive device that is energized by a passing train, and then communicates with the BTM (Balise transmission module) attached on the train via telegram messages to update the train of its location. However, there are vulnerabilities present in the balise air-gap interface that can be exploited by malicious attackers to alter or manipulated information on telegrams. This paper describes malicious attacks such as changing the locality information of balise can be exploited from attacking the vulnerability of air-gap interface of the balise-based system. This paper therefore proposes three security designs to check data integrity on telegram messages so as to protect the train balise telegram data integrity.

I. INTRODUCTION

CBTC (Communication based train control system) is an essential system in train and railway operation as it conducts the way trains in the railway move at certain speed and stop at stations [1]. It ensures that the traffic of the trains goes smoothly and without any incidents. It also ensures that train parks on the stations at the right time and for an appropriate time. ETCS (European Train Control System) is responsible for the signaling and control of the ERTMS (European Railway Traffic Management System) as controlling the movement of the train. To facilitate the ETCS, a balise and a Balise Transmission Module (BTM) will be required as shown in Fig. 1. A balise is a passive transponder and does not require battery power for operation. It transmits the stored data to a BTM on an oncoming train when the balise gets energized. Balises are located in between the rails of a railway and are located once every few meters. This is to ensure that the ETCS is aware of the train location and make the appropriated decisions on the movement and speed limits of the train [2].

BTM is a module that is used to receive telegram messages sent from the balises. BTM is attached under train and connects to an On-Board Unit in the train. The telegram sent from the balise and received by the BTM will be transferred to the on-board unit, which is responsible for controlling the movement of the train. Thus, the BTM and balise work together to manage train-ground communications and ensure the reliability and safety of train operations. Communication between the balise and BTM happens via the air-gap interface and it is important to constantly monitor the air-gap interface so as to prevent accidental errors [3].

Fig. 1. Balise and BTM

Accurate information from the balise is essential in ensuring that the train runs and parks at each station safely. If the location data from balise is altered or compromised, it will lead to serious consequences such as train emergency break, collision and derailment [4]. The failure of detecting a series of balises can also lead to such serious consequences. As the air gap between the balise and BTM lacks of security protection, telegrams sent by the balise are in plaintext and lack any integrity checks or any timestamps [5]. This makes it severely vulnerable to potential attacks and intrusion by potential attackers with malicious intents. Attacks such as tampering with telegram data, signal jamming, or balise displacing are such examples that can happen and compromise the safety of the people commuting by the train [5].

Attacks related to tampering with telegram data can be achieved in a few ways listed below:

- Faking of balise telegram data

Telegrams are sent in plaintext, which meant that it is readable data that is transmitted without encryption or “in the clear” [3]. This allows attackers can create a similar telegram and fake it as a telegram to be sent to passing trains. The BTM of the train will not be able to filter out the faked telegram and will accept the fake data as real information for the train to follow. For example, if the fake telegram that is sent by the attacker at a point near a sharp turn of the rail is being accepted by the BTM on the train, a derailment tragedy will happen if the attacker

*Research supported by the National Research Foundation (NRF), Prime Minister’s Office, Singapore.
Huaqun Guo and Jiqiang Lu are with Institute for Infocomm Research, A*STAR, Singapore 138632 (phone: +65 6408 2042; fax: +65 6776 1378; e-mail: guohq@i2r.a-star.edu.sg; lvjiqiang@hotmail.com).
Jonathan Zhi Wei Sim is with Department of Electrical & Computer Engineering, National University of Singapore, Singapore 599489 (e-mail: a01252558@e.ntu.edu.sg).
Bharadwaj Veeravalli is with Department of Electrical & Computer Engineering, National University of Singapore, Singapore 599489 (e-mail: elebv@nus.edu.sg).
chooses to send an instruction for the train to continue at full speed towards the sharp turn.

- **Attack via wireless communication in air-gap interface**
  Attackers can use a wireless emitter to disrupt the communication between the balise and BTM and compromise the delivery of the telegram to the passing train. This jamming attack however can be detectable by security cameras or by the railway workers in the vicinity.

- **Impersonation of BTM to collect telegram data**
  By using a device to impersonate as a BTM, attackers can receive the telegram and read it.

- **Cloning of data**
  Attackers might copy data from one balise and transfer it to another balise.

- **Displacement attack**
  Attackers can replay telegram of a balise at other location of the track, which is intended to create confusion for the train and control center [5]. For example, attackers can then be able to replay the telegram of a balise which indicates that it should run at normal high speed, on a balise near a curved rail section. This will lead to derailment of the train if it continues at a high speed towards the curved rail and lead to disaster consequences.

- **Direction reversal attack**
  As some rail sections on the railway are bi-directional, potential attackers can replay telegrams that have data of balise that are in reverse direction to the train, which can lead to misdirection of trains and possible collisions to happen [5].

In this paper, we study the train-ground communication vulnerabilities. We therefore propose security protection designs to ensure that only genuine balise data are communicated to the train and secure the communications between the balise and the train.

The remainder of this paper is organized as follows. Section II introduces the literature review. Section III presents our security protection designs to ensure the balise telegram data integrity. The experimental tests and performance are presented in Section VI. Finally, Section V outlines our conclusions.

## II. LITERATURE REVIEW

### A. ERTMS (European Railway Traffic Management System)

ERTMS is a system that uses communication-based signaling for the trains and defines how trains on a railway system communicate and operate. The ERTMS standard consists of two main components, the GSM-R and ETCS. The GSM-R is a radio system that enables communication between trains and RBC (Radio Block Center). The ETCS is an automatic control system that controls the movement of the train [6].

### B. ETCS (European Train Control System)

In ETCS, the eurobalise transmission system is responsible for the transmission of data from a ground-based balise to an antenna unit of a passing train. The data will then be received by a BTM and an on-board unit of a train, which controls the movement of the train and determines when it should stop (Fig. 2). The balise is a passive transponder that locally stores information such as its location on the rail and the speed restriction for passing trains. It is placed between the rails of a railway and does not require any battery to operate. It is used as a position marker and is way to control the movement of trains. It transmits the information to passing trains by getting telepowered by the BTM under the trains. The BTM is a module mounted under each train and energises the balise when it passes through the balise. The energized balise will then send a telegram to be received by the BTM. Communications between balise and BTM via telegram messages are crucial in ensuring the reliability and safety of the train operation [3].

![Fig. 2. Structure of Eurobalise transmission system [3]](image)

### C. Balise Telegram

The balise will transmit a telegram repeatedly to the BTM so long as it is within the communication range and gets tele-powered. This is to ensure that there will be at least one successfully transmitted telegram that is received in the BTM under a passing train. Each balise telegram will have a 14-bit balise group identity, a unique balise identity (baliseID) and various packet types that can be translated by the BTM. The balise telegram has two compatible lengths, 341 bits and 1023 bits, respectively. The structure of the telegram [4] is shown in Fig. 3.

![Fig. 3. Balise Telegram Data Format](image)

**Shaped data:** Contains the payload information that is scrambled and encoded to prevent transmission errors. Control bits: Presently contains the binary string 001b. Scrambling bits: Contains the initial state of the scrambler used on the data bits before the scrambling. Extra Shaping bits: Used to execute shaping constraint on the check bits independent of the scrambling. Checksum: Contains 75 parity bits for error correcting and 10 bits for synchronization purposes. The more details of information
of the structure of the balise telegram can be found on the EUROBALISE standard [3]. The ‘shaped data’ structure of a telegram is a packet which contains a header, end of information segment and a payload. The payload is the one where the main information is stored in.

D. Cybersecurity with Transportation Systems

The cybersecurity issues of transportation systems have received deserved attention in recent years. In August 2012, the U.S. Department of Homeland Security released its “Roadmap to Secure Control Systems in the Transportation Sector” [7], suggesting high-level strategies to improve the security standing of control systems in transportation systems. In June 2013, the American Public Transportation Association published its recommended practice for “Securing Control and Communications Systems in Rail Transit Environments” [8]. In 2013, the European Union launched a new call for the studies of cybersecurity in land transport [9].

However, some major questions remain open and unresolved. In particular, a train system is a complex coupling of heterogeneous subsystems, which altogether support continuous flows of a large number of passengers over geographically distributed areas. Such sophisticated and highly automated systems pose grand security challenges that have not been addressed collectively before. According to an investigation by Sky News, a UK multimedia news organization, the UK rail network has been hit by at least four major cybersecurity attacks over a 12-month period between the years 2015 and 2016 [10].

III. SECURITY PROTECTION FOR BALISE TELEGRAM

Balise telegram does not have any security guarantees and hence makes it impossible to detect if a data being transferred is false or has been tampered with. This is a serious concern as attackers with malicious intent can alter the data without being notice. Hence this paper focuses on addressing this issue to protect the balise telegram data integrity.

As seen in Fig 3, if the Shaped data is 913 bits for the long telegram, the payload portion of the Shaped data could be 830 bits in length [3]. The structure of the payload data in a balise telegram with location packet is shown in Fig. 4. The shaped data contains the locality of each unique balise that will be transmitted to the BTM on the train. In this payload data of a total length of 830 bits, the location packet has 232 bits of filled data and 598 bits of empty bits.

![Fig 4. Data structure of the payload data](image)

A. Design and Implementation of Data Integrity Checking

A design that is more secure and trusted would be the one that has integrity, authenticity and confidentiality if possible. By protecting the data integrity and authenticity, we can detect the compromised balise telegram or faking information being transmitted to passing trains. When coming up with the security protection design for the telegram messages being sent from balises to the train, both AES-CCM and HMAC approaches provide data integrity check for telegram messages received by the BTM from balises.

![Fig 5. Process for integrity checking with AES-CCM (Ciphertext)](image)
B. Data Integrity Checking with AES-CCM (Ciphertext)

AES-CCM is a mode of operation of block ciphers that combines both Cipher Block Chaining (CBC) and Message Authentication code (MAC) with the AES Cipher Block [11] and does encryption of the input message, authentication and integrity check of the message transmitted. It requires four inputs, namely an AES key, a nonce, a plaintext and optional additional authenticated data for add-ons. The output generated from AES-CCM is a cipher text and an authentication tag (MAC) [11]. The length of the mac or authentication tag from AES-CCM implementation can be the selected value from 4, 6, 8, 10, 12, 14, and 16. The authentication tag generated from AES-CCM will be placed in the empty bits section of the payload data structure. The data to be encrypted will be the location details in Packet 79 is from Q_SCALE (Last bit) till M-POSITION (k) (24bits) in Fig. 4. This data is 136 bits in length and is used to represent the plaintext to be encrypted with AES-CCM for this design.

Because the balise is passive, and hence nonce, cipher text and authentication tag are generated offline and embedded into balise telegram. The value of the baliseID, nonce and authentication tag is appended to the ciphertext before the finalized telegram message is embedded into the balise. This additional baliseID, nonce and authentication tag value will become a new packet that is appended together and put into the payload of telegram.

Fig. 5 shows the process for integrity check using AES-CCM, and detailed steps are listed below.

Step 1: A secret key, a generated nonce and plaintext are used as input, and then apply AES-CCM function to generate a 128-bit cipher text and an authentication tag.
Step 2: The 128-bit cipher text plus 8-bit of zero padding is replaced the original location Packet 79 of 136 bits.
Step 3: This additional baliseID, nonce and authentication tag value will become a new packet that is added to the empty bits of original telegram.
Step 4: The telegram message is embedded into the balise.
Step 5: The balise is energized by a passing train and transmits the telegram message to the BTM on the oncoming train.
Step 6: The BTM receives the telegram message and splits it to get four variables baliseID, nonce, ciphertext and authentication tag.
Step 7: The baliseID is used to get the corresponding unique secret key.
Step 8: The cipher text is decrypted with AES-CCM decryption.
Step 9: The plaintext from the decryption in the BTM is encrypted again to generate an authentication tag.
Step 10: Comparison is made between the received authentication tag and the authentication tag generated in the BTM. If both authentication tags are same, the telegram message is authenticated and passed the integrity check. Otherwise, the telegram message is not authenticated and is manipulated or fake, which will be discarded.

C. Data Integrity Checking with AES-CCM (Plaintext)

This sub-session presents another way of data integrity checking with AES-CCM without encryption. In this way, implementation of data integrity checking is only done at BTM without any changes on the balise. The BTM will store a fixed nonce value to be used as input for AES-CCM, and a unique secret key value for each unique balise into its internal database. The actual authentication tag value that is associated with each balise is also stored in the BTM internal database for comparison and data integrity check with incoming plaintext received by the BTM. The detailed steps to process integrity checking with AES-CCM (plaintext) are listed below.

Step 1: A telegram message with plaintext is sent from the balise to the BTM.
Step 2: From the plaintext, BTM will get the baliseID, and then its responding secret key, nonce, and authentication tag from the pre-stored internal database.
Step 3: BTM uses the received plaintext, the secret key and nonce as input to use AES-CCM encryption to generate an authentication tag.
Step 4: Comparison is made between the stored authentication tag and the authentication tag generated in the BTM. If both authentication tags are same, the telegram message is authenticated and passed the integrity check. Otherwise, the telegram message is not authenticated and is manipulated or fake, which will be discarded.

In this design, the telegram in the balise will not be changed and still be sent in plaintext to the BTM, and only the BTM will require additional process and storage for the data integrity checking. Due to the plaintext transmitted from the balise to the BTM, there is no confidentiality in this design.

D. Data Integrity Checking with HMAC

There is another design with HMAC, which can also achieve the data integrity checking. In this design, the balise and the BTM on the train share a secret key that undergoes the MAC function together with the plaintext message to generate an authentication tag. The generated authentication tag is inserted into the telegram packet. The BTM on the oncoming train will receive the telegram from the balise and verify the data integrity of the telegram by comparing the authentication tag sent from the balise with an authentication tag generated by the BTM. Similarly, Fig. 6 shows the process for integrity check with HMAC. And the detailed steps are listed below.

Step 1: A secret key and the plaintext are used as input, and then apply HMAC function to generate an authentication tag.
Step 2: This authentication tag value will become a new packet that is added to the empty bits of original telegram.
Step 3: The telegram message is embedded into the balise.
Step 4: The balise is energized by a passing train and transmits the telegram message to the BTM on the oncoming train.
Step 5: The BTM receives the telegram message and splits it to get the balanceID, plaintext and the authentication tag.

Step 6: The balanceID is used to get the corresponding unique secret key in the BTM.

Step 7: The plaintext received from the balance and the unique secret key in the BTM are used as input, and then apply HMAC function to generate an authentication tag.

Step 8: Comparison is made between the received authentication tag and the authentication tag generated in the BTM. If both authentication tags are same, the telegram message is authenticated and passed the integrity check. Otherwise, the telegram message is not authenticated and is manipulated or fake, which will be discarded.

![Figure 6. Process for integrity checking with HMAC](image)

**IV. Performance Evaluation**

This section will compare the three security protection designs for data integrity checking presented in the previous section.

A. Test-bed setup

The test-bed was designed to simulate the communication between a balance and a BTM on a train as well as the transmission of telegram from the balance to the BTM. Table I shows the list of equipment and software used in the testbed setup.

<table>
<thead>
<tr>
<th>TABLE I. ITEMS USED FOR TEST-BED SETUP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product</strong></td>
</tr>
<tr>
<td>Lenovo Laptop</td>
</tr>
<tr>
<td>Lenovo Laptop</td>
</tr>
<tr>
<td>Atom-IDE</td>
</tr>
<tr>
<td>Ubuntu terminal</td>
</tr>
</tbody>
</table>

The implementation is demonstrated with python programming and is displayed on the bash terminal. The python package pycryptodome 3.5.1 is used, which contains cryptographic primitives of HMAC. The total length of the telegram message will be 1023 bits. The length of the data to be encrypted in the ‘shaped data’ part of the data will be 830 bits as a long telegram is used.

B. Efficiency of Data Integrity Checking

With data integrity checking, there will be additional processing time that will occur. Hence the efficiency of Data Integrity Checking is tested. Table II shows the additional time needed in BTM to perform the data integrity checking with AES-CCM (CipherText). Table III shows the additional time needed in BTM to perform the data integrity checking with AES-CCM ( Plaintext), while Table IV shows the additional time needed in BTM to perform the data integrity checking with various MAC functions.

<table>
<thead>
<tr>
<th>TABLE II. ADDITIONAL TIME TAKEN IN BTM FOR AES-CCM (CIPHERTEXT)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product</strong></td>
</tr>
<tr>
<td>AES-CCM with 4-byte Tag</td>
</tr>
<tr>
<td>Average Time (ms)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE III. ADDITIONAL TIME TAKEN IN BTM FOR AES-CCM (PLAINTEXT)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product</strong></td>
</tr>
<tr>
<td>AES-CCM with 4-byte Tag</td>
</tr>
<tr>
<td>Average Time (ms)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE IV. ADDITIONAL TIME TAKEN IN BTM FOR HMAC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product</strong></td>
</tr>
<tr>
<td>AES-CCM with 4-byte Tag</td>
</tr>
<tr>
<td>Average Time (ms)</td>
</tr>
</tbody>
</table>

Tables II shows the greatest additional average time taken in AES-CCM (CipherText) to separate the ciphertext, balanceID, nonce, and authentication tag from the telegram, decryption of the ciphertext, generation of the authentication tag and comparison received authentication tag and generated authentication tag in the BTM and the range of additional average time is around 2.782 milliseconds to 2.991 milliseconds. From Table III, the additional average time taken in BTM for AES-CCM (PlainText) is a value of around 0.9215 milliseconds to 1.485 milliseconds. From Table IV, the additional average time taken in BTM for HMAC is a value of around 0.9216 milliseconds to 1.575 milliseconds. All three time values are very small and could meet the requirement for the range of time taken for the trackside CBTC equipment reaction (0.07 – 1 second) and train-borne CBTC equipment reaction (0.07-0.75 seconds).

This means that the time taken for the additional overhead to perform data integrity checking can be considered insignificant in affecting the time taken to transmit the telegram from the balance to the BTM, and add the value of data integrity. AES-CCM (CipherText) adds...
additional confidentiality to the telegram message being transmitted to the BTM.

C. Different Length of Authentication Tag

The AES-CCM Cipher allows for the authentication tag generated to have a length of 4, 6, 8, 10, 12, 14 or 16 bytes. Data with an authentication tag with a smaller byte size will then require less bits length in the telegram, less bandwidth to be transmitted, as well as lesser transmission time. However, using a tag smaller than 16 bytes increases the probability of the attacker creating a correct tag at random. Since the time difference is very little, it may be good to use 16 bytes for the authentication tag length.

D. Comparison of Three Designs

Three security designs to protect the train balise telegram data are compared and summarized into Table V.

<table>
<thead>
<tr>
<th></th>
<th>AES-CCM (Ciphertext)</th>
<th>AES-CCM (Plaintext)</th>
<th>HMAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Authenticity</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Integrity</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Append an authentication tag to the telegram message</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Store authentication tag in balise</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Length of authentication tag</td>
<td>small</td>
<td>small</td>
<td>large</td>
</tr>
<tr>
<td>Store authentication tag in BTM</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Additional time for integrity checking</td>
<td>Less than 3ms</td>
<td>Less than 2ms</td>
<td>Less than 2ms</td>
</tr>
</tbody>
</table>

From Table V, if only to protect the train balise telegram data integrity, AES-CCM (Plaintext) is the best choice. The reason is that this design does not make the changes on the original telegram messages and does not make the changes on the balises, which are a lot along rails. In addition, the length of authentication tag is smaller than the one with HMAC, which is from 16 bytes to 64 bytes. If the confidentiality is requested, then AES-CCM (Ciphertext) is a choice.

V. CONCLUSION

This paper analyzes and identifies the vulnerabilities with train balise telegram data which is communicated from the balise to the BTM on the train. The security designs to protect the train balise telegram data integrity are presented in details. The simulations are conducted to evaluate the additional time for the data integrity checking. From the performance results, the time taken for the additional time overhead to perform data integrity checking can be considered insignificant in affecting the time taken to transmit the telegram from the balise to the BTM. If only to protect the train balise telegram data integrity, AES-CCM (Plaintext) is the best choice because this design does not make the changes on the original telegram message and the balise, and the length of authentication tag is smaller than the one with HMAC. If the confidentiality is requested, then AES-CCM (Ciphertext) is a choice.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation (NRF), Prime Minister’s Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2014NCR-NCR001-31) and administered by the National Cybersecurity R&D Directorate. The special thanks are also given to SMRT Trains Ltd to provide domain knowledge and technical support.

REFERENCES